RBniCS – reduced order modelling in FEniCS

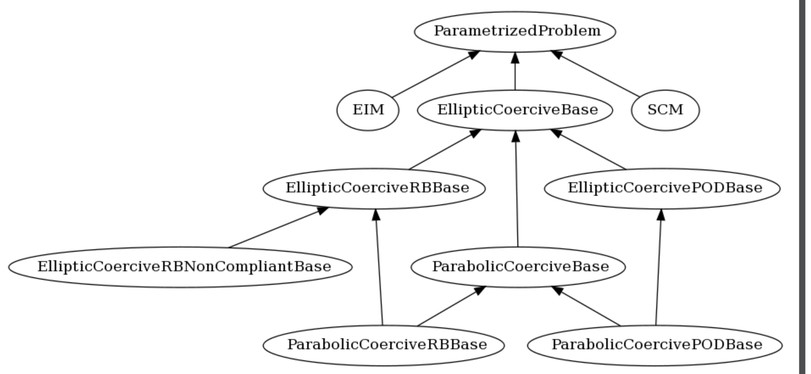
F. Ballarin, A. Sartori, G. Rozza
SISSA mathLab, International School for Advanced Studies, Trieste, Italy

Abstract

RBniCS [1] is a python-based library, developed on top of FEniCS [3], aimed at the developement of reduced order models in the FEniCS environment. In particular, reduced order techniques such as the **certified reduced basis method** and **proper orthogonal decomposition-Galerkin** methods are implemented. The FEniCS project allows RBniCS to take advantage of the high-level (e.g., human readable) code used for the automated solution of partial differential equations. Thanks to the features of FEniCS the final user needs to prepare a **short code** to carry out a reduced order simulation.

It is ideally suited for **novice users** willing to learn reduced basis methods and reduced order modelling, thanks to an **object-oriented approach** and an intuitive and versatile python interface. Indeed, it is a companion of the introductory reduced basis handbook [2], and has been already used in doctoral classes within the "Mathematical Analysis, Modelling, and Applications" PhD course at SISSA, as well as for courses within the "Master in High Performance Computing" jointly offered by SISSA and International Centre for Theoretical Physics (ICTP).

Object-oriented approach

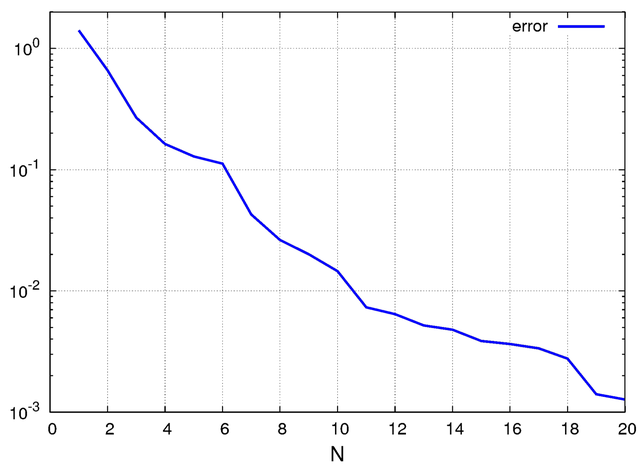


Template of a main program

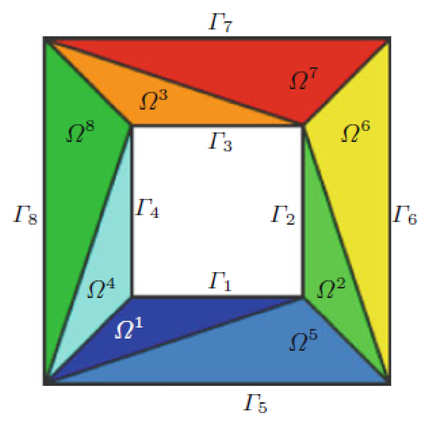
```
from dolfin import *
from RBniCS import *
mesh = Mesh("mesh.xml")
subd = MeshFunction("size_t", mesh,
    "mesh_physical_region.xml")
bound = MeshFunction("size_t", mesh,
    "mesh_facet_region.xml")
V = FunctionSpace(mesh, "Lagrange", 1)
tb = MyProblem(V, subd, bound)
mu_range = [(0.1, 1.0), (1.0, 10.0)]
tb.setmu_range(mu_range)
tb.setxi_train(1000)
tb.setNmax(25)
tb.offline()
online_mu = (0.5,6.0)
tb.setmu(online_mu)
tb.online_solve()
tb.setxi_test(500)
tb.error_analysis()
```

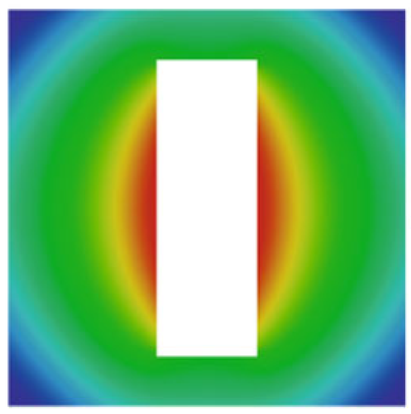
Tutorials 1 and 2: basics

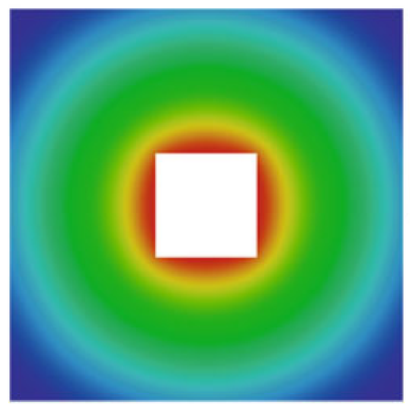
```
class MyProblem(EllipticCoerciveRBBase):
  def __init__(self, V, subd, bound):
      ... # constructor ...
  def compute_theta_a(self):
      mu1 = self.mu[0]
      return (mu1, 1.)
  def compute_theta_f(self):
      mu2 = self.mu[1]
      return (mu2,)
  def assemble_truth_a(self):
      u = self.u; v = self.v; dx = self.dx
      # Assemble
      a0 = inner(grad(u),grad(v))*dx(1)
      A0 = assemble(a0)
      a1 = inner(grad(u),grad(v))*dx(2)
      A1 = assemble(a1)
      # Return
      return (A0, A1)
  def assemble_truth_f(self):
      v = self.v; ds = self.ds
      # Assemble
      f0 = v*ds(1); F0 = assemble(f0)
      # Return
      return (F0,)
  def get_alpha_lb(self):
      return min(self.compute_theta_a())
```



Tutorials 3 and 4: geometrical parametrization

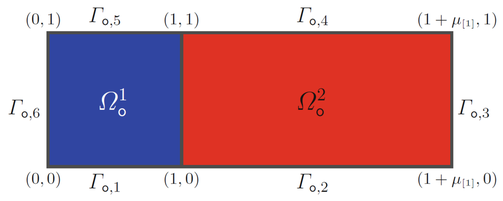


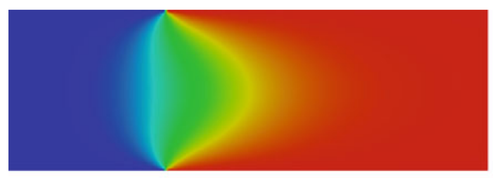


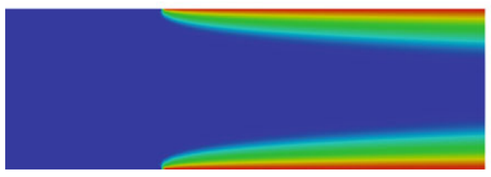


Tutorial 4: stability factor by SCM

```
class MyProblemWithSCM(EllipticCoerciveRBBase):
  def __init__(self, V, subd, bound):
      ...
      # Initialize an SCM object
      self.SCM_obj = SCM(self)
  def offline(self):
      # Perform first the SCM offline phase, ...
      bak_first_mu = tuple(list(self.mu))
      self.SCM_obj.offline()
      # ..., and then call the parent method.
      self.setmu(bak_first_mu)
      EllipticCoerciveRBBase.offline(self)
  def get_alpha_lb(self):
      return self.SCM_obj.get_alpha_LB(self.mu)
```

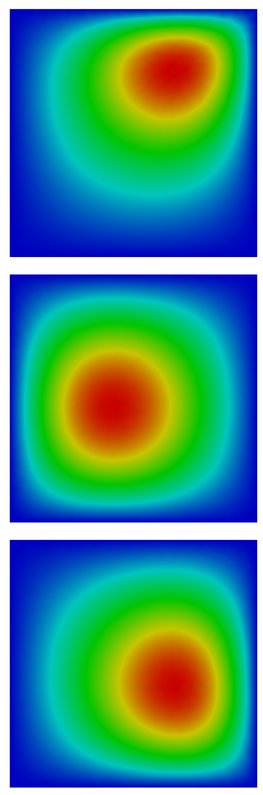






Tutorial 5: affinity assumption recovery by EIM

```
class MyProblemWithEIM(EllipticCoerciveRBBase):
  def __init__(self, V, subd, bound):
      ...
      # Initialize an EIM object
      self.EIM_obj = EIM(self)
      self.EIM_obj.parametrized_function = "exp( -2*pow(x[0]-mu_1, 2)
          -2*pow(x[1]-mu_2, 2) )"
  def compute_theta_f(self):
      self.EIM_obj.setmu(self.mu)
      return self.EIM_obj.compute_interpolated_theta()
  def assemble_truth_f(self):
      v = self.v; dx = self.dx
      # Call EIM
      self.EIM_obj.setmu(self.mu)
      interpolated_gaussian = self.EIM_obj.
          assemble_mu_independent_interpolated_function()
      # Assemble
      all_F = ()
      for q in range(len(interpolated_gaussian)):
          f_q = interpolated_gaussian[q]*v*dx
          all_F += (assemble(f_q),)
      # Return
      return all_F
```



How to get RBniCS

RBniCS is freely available under the GNU LGPL, version 3, at

http://mathlab.sissa.it/rbnics

References

[1] F. Ballarin, A. Sartori, and G. Rozza. RBniCS – reduced order modelling in FEniCS. http://mathlab.sissa.it/rbnics, 2015.

[2] J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. SpringerBriefs in Mathematics. Springer, 2015.

[3] A. Logg, K.-A. Mardal, G. N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

Acknowledgements

This work has been supported by the project INDAM-GNCS 2015, "Computational Reduction Strategies for CFD and Fluid-Structure Interaction Problems" and by the PRIN project "Mathematical and numerical modelling of the cardiovascular system, and their clinical applications".